File No. 7090-48
Form C28-6386-1

IBM Systems Reference Library

IBM 7090/7094 IBSYS Operating System

Version 13

Symbolic Update Program

This publication describes the capabilities of the Symbolic Update Program (7090-UT-978), the requirements for its implementation, and the pseudo-instructions it uses. This program operates under control of the IBM 7090/7094 IBSYS Operating System, Version 13, and allows users to modify serialized symbolic tapes, including those for the operating system itself.

PREFACE

This publication describes the capabilities of the IBM 7090/7094 Symbolic Update Program, the requirements for its implementation, and the pseudo-instructions it uses.

Material in this publication is intended for programmers experienced in the use of the IBM 7090/7094 IBSYS Operating System. Readers unfamiliar with this system are referred to the publication IBM 7090/7094 IBSYS Operating System, Version 13: System Monitor (IBSYS), Form C28-6248. Other publications that may be helpful are listed in the publication IBM 7090/7094 Bibliography, Form A28-6306.

The Symbolic Update Program requires the following machine configuration:

An IBM 7090 or 7094 Data Processing System.

An IBM 716 Printer.

An IBM 729 Magnetic Tape Unit, an IBM 1301 Disk Storage Unit, or an IBM 7320 Drum Storage Unit, for system residence.

Four IBM 729 Magnetic Tape Units, one each for system input, system output, update input, and update output.

MAJOR REVISION (January 1965)

This publication, Form C28-6386-1, is a major revision of the previous edition, Form C28-6386-0, and makes that publication obsolete. The previous edition has been extensively revised and rewritten. New material has been added. Included in this new material are a new pseudo-instruction, a new option for the $TITLE card, and a new section on serializing unserialized decks.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A form for readers' comments appears at the back of this publication. It may be mailed directly to IBM. Address any additional comments concerning this publication to the IBM Corporation, Programming Systems Publications, Department D39, 1271 Avenue of the Americas, New York, N. Y., 10020

CONTENTS

GENERAL DESCRIPTION 5

Capabilities 5

Similarity to FAP Update 5

Tapes Used 5

Control Cards 6
Required Control Cards 6
Optional Control Cards 7

OPERATION OF UPDATE 9

Inserting New Instructions 9

Using Update Pseudo-Instructions . . . 9
Card Format 9
UPDATE Pseudo-Instruction 9
NUMBER Pseudo-Instruction 10
====== Pseudo-Instruction 11
DELETE Pseudo-Instruction 12
IGNORE Pseudo-Instruction 12
SKIPTO Pseudo-Instruction 12
ENDFIL Pseudo-Instruction 13
REWIND Pseudo-Instruction 13
UNLOAD Pseudo-Instruction 14
SKPFIL Pseudo-Instruction 14
PRINT Pseudo-Instruction 14
ENDUP Pseudo-Instruction 14

Serialization 15
Sequence Checking 15
Optional Serialization 15
Serializing Unserialized Decks . . 15

Tape Positioning 16

Illegible Input Instructions 16

PROGRAMMING EXAMPLES 17

APPENDIX A: TAPE UNIT ASSIGNMENTS . . 20

APPENDIX B: BINARY COLLATING SEQUENCE 21

INDEX 22

FIGURES

Figure 1. Tapes Used for an Update Job 6
Figure 2. Update Input Tape 18
Figure 3. System Input Tape 18
Figure 4. System Output Tape 19
Figure 5. Update Output Tape 19
Figure 6. Unit Selection Guide . . . 20
Figure 7. Binary Collating Sequence . 21

The IBM 7090/7094 Symbolic Update Program (Update) is a subsystem of the IBSYS Operating System and is used to update symbolic tapes by changing, deleting, or adding symbolic card images and producing a new symbolic tape.

CAPABILITIES

Update may be used to modify any tape written in BCD mode, provided columns 73-80 of card images are available for serialization. The program recognizes serialization only in columns 73-80. Tapes that are serialized in other columns must also be serialized in columns 73-80 before updating. For example, a COBOL program on tape which is serialized in columns 1-6, must also be serialized in columns 73-80 before it is updated. Update can be used to maintain multireel input and output tapes; subdivide or extract card images from input tape; space tape; and check the sequence of serialization.* In addition, it can be used to maintain subsystems of the IBSYS Operating System and the operating system itself. For information on updating the IBSYS Operating System, see IBM 7090/7094 IBSYS Operating System: System Monitor (IBSYS), Form C28-6248.

The Symbolic Update Program can read symbolic tapes containing 14 words per card, either unblocked or blocked up to 16 cards per block. Normally, it produces a blocked tape--14 words per card, 10 cards per block--with System Monitor, IBJOB Processor, and END cards unblocked. A control-card option is available to produce an unblocked output tape.

SIMILARITY TO FAP UPDATE

The Symbolic Update Program is similar to the update-only mode of the IBM 7090/7094 FORTRAN II Assembly Program (FAP). Decks previously prepared for the FAP update program may be used with Update merely by specifying UPDATE in the variable field of a $EXECUTE card that precedes the update cards.

However, the UMC pseudo-operation of FAP has no counterpart in Update and, if used, will be ignored. An *FAP card will also be ignored. Note also that Update contains only updating facilities. Thus, if compilation or assembly of a changed program is desired, it must be done after completion of updating, using a program such as the IBJOB Macro Assembly Program (IBMAP), the FORTRAN IV Compiler (IBFTC), or the COBOL Compiler (IBCBC).

TAPES USED

The Symbolic Update Program requires five tape units: two for input, two for output, and one for system residence (see Figure 1). A direct access storage unit may be substituted for the last tape unit.

The system input tape contains the programmer's update job and its control cards. This tape may also contain other jobs for IBSYS subsystems.

The programmer's update job consists of (1) the control cards that call and control the operation of the Symbolic Update Program, (2) pseudo-instructions that specify the desired updating operations, and (3) any symbolic cards that are to be inserted in the update output tape.

The update input tape is the symbolic tape that requires updating. Any card image on this tape that is to be operated upon by Update must be serialized in card columns 73-80, and the last card image must be properly serialized. File marks on this tape are ignored.

The update output tape is a blocked or unblocked symbolic tape containing an updated version of the contents of the update input tape. It may be assembled or compiled later if it contains the necessary control cards.

The system output tape contains an annotated list of deleted and inserted instructions.

---

*The IBM 7090/7094 Symbolic Update Program uses the binary collating sequence, shown in Appendix B.

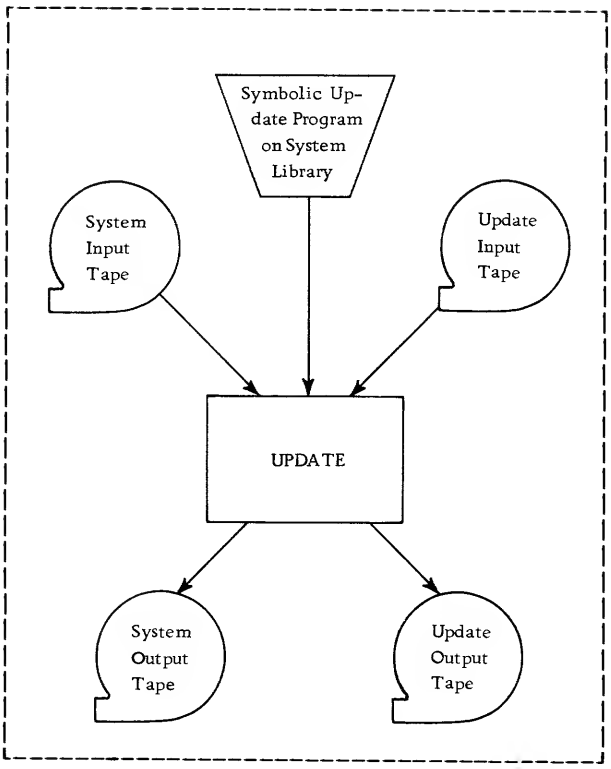

Figure 1. Tapes Used for an Update Job

System input is on the system input unit (SYSIN1)* and system output is on the system output unit (SYSOU1). Other functions may be on any available system unit. (See "Appendix A.")

CONTROL CARDS

An update job requires a $JOB card and a $EXECUTE card with UPDATE specified in the variable field. These cards are described below under "Required Control Cards." Several other control cards are recognized by Update and may also be used. These cards are described below under "Optional Control Cards."

Control cards are recognized any place except between an UPDATE pseudo-instruction and its matching ENDUP pseudo-instruction. Control cards appearing between the UPDATE and ENDUP pseudo-instructions are treated as insertion cards.

---

*The card reader may not be used as the system input unit for the Symbolic Update Program.

REQUIRED CONTROL CARDS

The following control cards are required when using the Symbolic Update Program:

$JOB Card

The $JOB card defines the beginning of a job. The format of the $JOB card is:

```
1               16
$JOB            any text
```

This control card causes control to be transferred to an installation accounting routine if there is one and the restoration of any units that were reassigned or made available during a previous job except:

Any unit logically detached by IBSYS.

Any unit assigned to a system unit function in place of a detached unit.

Any unit that was assigned to a system input, system output, or system peripheral punch function.

If a $JOB card appears when the Symbolic Update Program is in control, the System Supervisor is called into core storage if it is required either to restore the status of a unit or to control a manually initiated between-jobs interrupt condition. Then control is returned to Update.

A $JOB card is listed both on- and off-line. Columns 16 through 72 are normally used to identify the job and may contain any combination of alphameric characters and blanks.

$EXECUTE Card

The $EXECUTE card defines the beginning of a job segment to be processed by a subsystem, in this case the Symbolic Update Program. The format of the $EXECUTE card is:

```
1               16
$EXECUTE        UPDATE
```

When this card is read by the System Supervisor, it reads in the first record of Update and transfers control to that subsystem. If a $EXECUTE card spec-

ifying a subsystem other than UPDATE appears while Update is in control, the subsystem returns control to IBSYS.

OPTIONAL CONTROL CARDS

Update recognizes the following control cards:

$IBSYS Card

The $IBSYS card returns control to the System Monitor. The format of the $IBSYS card is:

```
1
$IBSYS
```

Update prints the message "RETURNING TO IBSYS" on-line and relinquishes control to the System Monitor.

$ID Card

Update transfers control to the installation accounting routine when a $ID card is read. If there is no accounting routine, no action occurs and the card is printed on-line. The format of the $ID card is:

```
1               16
$ID             any text
```

$STOP Card

Update indicates to the System Monitor that a $STOP card has been read and transfers control to IBSYS to process the card. The format of the $STOP card is:

```
1
$STOP
```

$PAUSE Card

The $PAUSE card causes a machine halt, and the contents of the card are printed on-line. To resume processing, the operator should press START. The format of the $PAUSE card is:

```
1               16
$PAUSE          instructions to operator
```

The variable field of this control card should contain an explicit message to the operator.

$* Card

The $* card is used as a comments card. The format of the $* card is:

```
1 3
$* any text
```

This card causes no action. It is merely listed on- and off-line, and Update continues.

$TITLE Card

The first line of each page of the system output listing contains a page number (starting with 1 for each update job) and the date (from SYSDAT in the IBSYS Monitor Nucleus). A $TITLE card may be used to suppress dating and to insert a title in this line. The format of the $TITLE card is:

```
1       8       16
$TITLE [NODAT] any text
```

If this card is used, it must appear after the $EXECUTE UPDATE card. It may not appear between UPDATE and ENDUP pseudo-instructions. If it appears between these pseudo-instructions, it is treated as an insertion card.

If NODAT appears in columns 8-12, no date appears in the system output listing. If columns 8-12 are blank, the date is listed.

The contents of columns 16-72 appear at the top of each page of the system output listing. A title may be changed by using another $TITLE card. For example, if three card decks were to be updated and a different title were desired for the system output listing for each deck, the following sequence would be used:

```
$JOB
$EXECUTE        UPDATE
$TITLE          TITLE FOR DECK 1
       UPDATE   2,3
       .
       .        (deck 1 changes)
       .
       ENDUP
$TITLE          TITLE FOR DECK 2
       UPDATE   2,3
       .
       .        (deck 2 changes)
       .
       ENDUP
$TITLE          TITLE FOR DECK 3
       UPDATE   2,3
       .
       .        (deck 3 changes)
       .
       ENDUP
```

Except for $TITLE, all of the preceding cards are IBSYS control cards and are further described in the publication IBM 7090/7094 IBSYS Operating System: System Monitor (IBSYS), Form C28-6248.

The update input tape is scanned once. During this scan, new instructions can be inserted, instructions can be deleted, tape can be manipulated, and serial numbers can be changed using update pseudo-instructions. Update scans the entire card image on the update input tape. If an update pseudo-instruction appears in columns 8-13 of a card image on the update input tape, the corresponding pseudo-instruction is performed.

Each card on the system input tape is matched with the first card on the update input tape with the same serial number. If there is no match, the update input tape is positioned before the first instruction of higher serialization. If there is a matching card on the update input tape, it is deleted. Then the card on the system input tape is interpreted. If it contains an update pseudo-instruction, the pseudo-operation is performed. If it contains anything else, its contents are substituted for the update input tape card image having the same serial number. If there is no match, the card image will be inserted at that point. The modified program, reserialized if requested, is written out on the update output tape.

INSERTING NEW INSTRUCTIONS

To insert new card images, serialize them and place them on the system input tape.

A serialized instruction on the system input tape replaces an instruction with matching serialization on the update input tape; an instruction with nonmatching serialization is inserted in sequence. Unserialized instructions are inserted immediately upon being encountered.

If only the first card of a group of instructions on the system input tape is serialized, the entire group is inserted. The inserted group can be numbered with the NUMBER pseudo-instruction.

Card images that are inserted or replaced appear, so labeled, on the system output tape.

USING UPDATE PSEUDO-INSTRUCTIONS

Update pseudo-instructions differ from many other pseudo-instructions, such as those in the MAP language, in that they are not assembled and, therefore, do not generate any new coding. They cause updating operations to be performed by providing parameters for the portions of the Symbolic Update Program that actually manipulate tapes, delete card images, etc.

The section "Programming Examples" illustrates the use of some update pseudo-instructions.

CARD FORMAT

Update pseudo-instructions are punched, one per card, in the following format: the name field, which may be blank, occupies card columns 1-6. Update ignores the name field of all pseudo-instructions except the NUMBER pseudo-instruction. Card column 7 is always blank. The operation field begins in column 8 and is five or six characters long. The variable field starts at column 16 and may extend through column 72. The variable field should be followed by a blank to separate it from the comments field. The comments field extends to column 72 and, in the absence of a variable field, may begin in column 17. Columns 73-80 are used for identification and serial numbering. Serialization is required with the DELETE, IGNORE, and SKIPTO pseudo-instructions. Serialization is optional with all other pseudo-instructions. The programmer should read "Optional Serialization" before serializing these other pseudo-instructions.

UPDATE PSEUDO-INSTRUCTION

The UPDATE pseudo-instruction assigns update input and update output tapes, and specifies whether blocking is required. The first pseudo-instruction of any update program must be UPDATE, and all other update pseudo-instructions must be used within the range of an UPDATE pseudo-instruction (i.e., between

UPDATE and ENDUP pseudo-instructions). The format of the UPDATE pseudo-instruction card is:

Name Field	blank
Operation Field	UPDATE
Variable Field	one or two logical tape numbers and a blocking subfield, all separated by commas
Serialization	optional

The first subfield of the variable field is the logical tape number of the update input tape, i.e., the tape containing the program to be updated. If there is no update input tape, this subfield should be null or zero. The second subfield is the logical tape number of the update output tape, i.e., the tape that is to contain the updated program. Tapes 1, 5, 6, and 7 may not be used as update tapes. (See "Appendix A" for information on tape unit assignments.) If the update output tape is to be unblocked, the third subfield must contain a character other than zero or 10. If this subfield is null, zero, or 10, the update output tape is blocked 10 cards per block. END cards and any card with a $ in column 1 (control cards) are unblocked. If the third subfield is null, the second subfield need not be followed by a comma.

If a fourth subfield is present, as was possible in the FORTRAN II Assembly Program (FAP), this subfield is ignored.

NUMBER PSEUDO-INSTRUCTION

The NUMBER pseudo-instruction is used to reserialize columns 73-80 of the card images on the update output tape. The format of the NUMBER pseudo-instruction card is:

Name Field	up to six alphameric characters or blank
Operation Field	NUMBER
Variable Field	a number in the first subfield and, optionally, a number in the second subfield
Serialization	optional

The serialization that this pseudo-operation generates in columns 73-80 consists of two parts: an alphameric constant and a numeric variable. The name field specifies the alphameric constant and the variable field specifies the numeric variable.

The alphameric constant may have from one to six alphameric characters, which are left-justified in columns 73-78. If no alphameric constant is specified (name field blank), the columns from 73 to the first digit of the numeric variable are filled with numeric zeros.

The product of the first and second subfields of the variable field specifies the first number in the sequence of numeric variables. The second subfield specifies the serializing increment, i.e., the number added to each numeric variable to produce the next numeric variable. Numeric variables are right-justified in columns 75-80. Unused columns between the alphameric constant and the numeric variable are filled with numeric zeros. For example,

```
AR   NUMBER 10,5
```

produces the following serialization in columns 73-80 on the update output tape:

```
AR000050
AR000055
AR000060
etc.
```

If the second subfield of the variable field is omitted, a serializing increment of 10 is assumed and the first subfield of the variable field is multiplied by 10 to produce the first numeric variable. For example,

```
AL   NUMBER 1
```

produces the following serialization:

```
AL000010
AL000020
AL000030
etc.
```

To serialize from zero an explicit zero must be placed in the first subfield so that the product of the first and second subfields is zero. If a zero is placed in the second subfield, serialization starts at zero but no incrementing occurs, i.e., serialization on all card images on the update output tape are the same. If the variable field is omitted, serialization is suspended regardless of the contents of the name field.

Care must be exercised to avoid overlapping of the alphameric constant and the numeric variable. Unpredictable serialization may result if such overlapping occurs. A carry from the numeric variable into the alphameric constant may also result in unpredictable serialization. Overlapping and carries can be avoided by allowing enough columns for the numeric variable to accommodate the anticipated serialization.

====== PSEUDO-INSTRUCTION

The ====== (Equal Sign) pseudo-instruction is used to change the spelling of update pseudo-instructions in order to avoid conflict with insertion cards having update pseudo-instructions to be inserted. The format of the ====== pseudo-instruction card is:

Name Field	blank
Operation Field	======
Variable Field	blank; or an update pseudo-instruction (except DELETE, IGNORE, SKIPTO, and ======) and an alternate spelling separated by a comma
Serialization	optional

The first subfield of the variable field specifies the update pseudo-instruction to be changed. The second subfield specifies the alternate spelling to be inserted in the update pseudo-instruction dictionary. An alternate spelling is a string of one to six alphabetic or numeric characters, at least one of which is alphabetic. No special characters are permitted except periods. When Update encounters the specified alternate spelling, the appropriate pseudo-operation is executed. When the standard spelling for that pseudo-instruction is encountered, it is not recognized and the card is treated as an insertion card.

An equal sign pseudo-instruction with a blank variable field cancels the effect of all previous equal sign pseudo-instructions. If not cancelled in this way, equal sign pseudo-instructions remain in effect until the next $IBSYS or $EXECUTE UPDATE card is encountered (outside the range of an UPDATE pseudo-instruction).

Following are the restrictions on using the equal sign pseudo-instruction:

1. It may not be used to change the spelling of itself or the DELETE, IGNORE, or SKIPTO pseudo-instructions.

2. Only one alternate spelling for a pseudo-instruction can be active at a given time. For example, the following sequence is <u>not</u> permitted:

```
====== PRINT,PRYNT
====== PRINT,WRITE
```

If this sequence is used, an off-line error message is printed for the second equal sign card and a card with WRITE in the operation field is treated as an insertion card. However, an alternate spelling may be redefined. For example, the following sequence is permitted:

```
====== PRINT,PRYNT
 .
 .
 .
====== PRYNT,WRITE
 .
 .
 .
====== WRITE,PRINT
```

In this example, the second equal sign card cancels the PRYNT spelling and establishes WRITE as the pseudo-instruction for the PRINT pseudo-operation. The third equal sign card cancels the WRITE spelling and re-establishes the standard spelling, PRINT.

3. A maximum of eight equal sign pseudo-operations may be used. This includes both original definitions and

redefinitions. If a ninth equal sign pseudo-instruction (except with a blank variable field) appears, it is not executed and processing continues.

4. Care must be exercised in changing the spelling of the UPDATE pseudo-instruction. The spelling of the UPDATE pseudo-instruction may be changed; however, if additional UPDATE pseudo-instructions are to follow the ENDUP pseudo-instruction, the standard spelling must be restored before the next ENDUP pseudo-instruction.

DELETE PSEUDO-INSTRUCTION

The DELETE pseudo-instruction causes deletion of one or more card images on the update input tape. Deleted cards will appear on the system output tape, labeled as deleted. The format of the DELETE pseudo-instruction card is:

Name Field	blank
Operation Field	DELETE
Variable Field	blank or THRU
Serialization	required for a delete operation

If the variable field is blank, the card image with matching serialization is deleted; if the variable field contains THRU, all card images starting at the current position of the update input tape, up to and including the card image with matching serialization, are deleted. In the latter case, if matching serialization does not exist, deletions are made up to, but not including, the next card image of higher serialization.

If the variable field is blank and no matching serialization exists, the update input tape is positioned at the next card image of higher serialization, but no deletion occurs.

Serialization is required in order to perform a delete operation. A DELETE card with blank serialization has no effect except to inhibit sequence checking of the next serialized card image.

IGNORE PSEUDO-INSTRUCTION

The effect of the IGNORE pseudo-instruction is identical to the effect of the DELETE pseudo-instruction, except that the cards it causes to be deleted do not appear on the system output tape. The format of the IGNORE pseudo-instruction card is:

Name Field	blank
Operation Field	IGNORE
Variable Field	blank or THRU
Serialization	required for an ignore operation

If the variable field is blank, the card image with matching serialization will be deleted; if the variable field contains THRU, all card images starting at the current position of the update input tape, up to and including the card image with matching serialization, are deleted. In the latter case, if matching serialization does not exist, deletions are made up to, but not including, the next card image of higher serialization.

If the variable field is blank and no matching serialization exists, the update input tape is positioned at the next card image of higher serialization, but no deletion occurs.

Serialization is required for an ignore operation. An IGNORE card with blank serialization has no effect except to inhibit sequence checking of the next serialized card image.

SKIPTO PSEUDO-INSTRUCTION

The SKIPTO pseudo-instruction causes deletion of one or more card images on the update input tape up to, but not including, a card with matching serialization. Deleted cards will not appear on the system output tape. The format of the SKIPTO pseudo-instruction card is:

Name Field	blank
Operation Field	SKIPTO
Variable Field	blank
Serialization	required for a skipping operation

All card images, starting at the current position of the update input tape, up to but not including the card image with matching serialization, are deleted. If no matching serialization exists, deletion continues indefinitely.

At the end of a SKIPTO, the update input tape is positioned at the card image with matching serialization.

This pseudo-operation differs in two ways from the IGNORE pseudo-operation with THRU in the variable field: the card with matching serialization is not deleted, and a card image with higher serialization does not terminate the operation.

Serialization is required for a skipping operation. A SKIPTO card with blank serialization has no effect except to inhibit sequence checking of the next serialized card image.

ENDFIL PSEUDO-INSTRUCTION

The ENDFIL (End of File) pseudo-instruction is used to write a file mark on the addressed update tape. Update will not transfer file marks from the update input tape to the update output tape. File marks on the update input tape must be inserted at the proper location on the update output tape with an ENDFIL pseudo-instruction. The format of the ENDFIL pseudo-instruction card is:

Name Field	blank
Operation Field	ENDFIL
Variable Field	blank or logical tape number
Serialization	optional

If the variable field is blank, Update assumes that a file mark is to be written on the update output tape; however, the logical tape number of the update output tape may be placed in the variable field if desired. In either case any partial block of instructions in the output buffer is written on the update output tape before the file mark is written. No data is moved from the output buffer if a logical tape number other than that of the update output tape is used. Logical tapes 1, 5, 6, and 7 may not be addressed by this pseudo-instruction.

Care should be exercised to assure that the file mark is written in the desired location. For example, assume that a 500-card deck is being updated and the last insertion is card 250. If the ENDFIL pseudo-instruction were placed immediately after the last insertion card, the file mark would appear after card 250 on the update output tape. To assure that the file mark appears after card 500 on the update output tape, a spacer card which is a duplicate of card 500 should be placed before the ENDFIL card.

REWIND PSEUDO-INSTRUCTION

The REWIND pseudo-instruction causes the addressed update tape to be rewound. The format of the REWIND pseudo-instruction card is:

Name Field	blank
Operation Field	REWIND
Variable Field	blank or logical tape number
Serialization	optional

If the variable field is blank, Update assumes that the current update output tape is to be rewound. If the variable field is blank or contains the logical tape number of the current update output tape, any partial block of instructions in the output buffer is written on the update output tape before the tape is rewound. If any other logical tape number appears in the variable field, no data is transferred from the output buffer before the tape is rewound. Any tape that is rewound is logically disconnected, and no Update operation referring to it can be executed unless the tape is reassigned by a subsequent UPDATE pseudo-instruction.

Tapes 1, 5, 6, and 7 may not be addressed by this pseudo-instruction.

UNLOAD PSEUDO-INSTRUCTION

The UNLOAD pseudo-instruction causes the addressed update tape to be rewound and unloaded. The format of the UNLOAD pseudo-instruction card is:

Name Field	blank
Operation Field	UNLOAD
Variable Field	blank or logical tape number
Serialization	optional

This pseudo-operation operates in the same way as the REWIND pseudo-operation except that the tape is unloaded after being rewound. The operator should be notified (use the PRINT pseudo-instruction) to mount a tape before the unit is reassigned by a subsequent UPDATE pseudo-instruction.

Tapes 1, 5, 6, and 7 may not be addressed by this pseudo-instruction.

SKPFIL PSEUDO-INSTRUCTION

The SKPFIL (Skip File) pseudo-instruction causes the addressed update tape to be spaced forward until a file mark is passed. The format of the SKIPFIL pseudo-instruction card is:

Name Field	blank
Operation Field	SKPFIL
Variable Field	blank or logical tape number
Serialization	optional

If the variable field is blank, the update input tape is spaced forward past the next file mark. If the logical tape number of the current update output tape is placed in the variable field, any partial block of instructions in the output buffer is written on the update output tape before an attempt is made to space forward beyond the next file mark.

Tapes 1, 5, 6, and 7 may not be addressed by this pseudo-instruction.

If an update input tape contains a binary file to be skipped, the SKPFIL pseudo-instruction should appear outside the range of the UPDATE pseudo-instruction that pertains to that tape.

This will prevent a redundancy error message from occurring, because an UPDATE pseudo-instruction initiates reading of the update input tape into the input buffer.

For example, to skip over a binary file on logical tape 8, use the following sequence:

```
UPDATE  ,9
SKPFIL  8
UPDATE  8,9
```

PRINT PSEUDO-INSTRUCTION

The PRINT pseudo-instruction first causes columns 14-72 to be printed on-line; then it causes a machine halt. The format of the PRINT pseudo-instruction card is:

Name Field	blank
Operation Field	PRINT
Variable Field	any alphameric characters (may start in col. 14)
Serialization	optional

The operator should press START to continue the job.

ENDUP PSEUDO-INSTRUCTION

The ENDUP (End Update) pseudo-instruction terminates an update job. The format of the ENDUP pseudo-instruction card is:

Name Field	blank
Operation Field	ENDUP
Variable Field	blank
Serialization	optional

Any card images in the output buffer are written on the update output tape before the job is terminated. A spacer card may be required before the ENDUP card to assure that the job is terminated at the proper location (see "Tape Positioning").

SERIALIZATION

This section describes sequence checking of serialized card images on the system input tape and the effects of providing optional serialization on certain pseudo-instruction cards. In addition, a method for serializing unserialized decks is described.

SEQUENCE CHECKING

Card images on the system input and update input tapes are checked for proper sequencing based on the serial numbers in card columns 73-80. If the cards are out of sequence, an error message is printed on-line and off-line and processing continues.

If columns 75-80 are blank, serialization in 73-80 is taken to be all zeros for sequencing. In this case, no warning is given for a sequence error, even though the card is taken as lower serialization than the one preceding it. Also, no matter what the serialization of the next serialized card, no sequence error occurs, because that card is necessarily higher than the one with zero serialization.

The update output tape may be reserialized by using the NUMBER pseudo-instruction. This provides proper serialization for subsequent assembly, compilation, or updating.

OPTIONAL SERIALIZATION

Only the DELETE, IGNORE, and SKIPTO pseudo-instructions require serialization in card columns 73-80 for normal operation. Without serialization, they have no effect except to inhibit sequence checking of the next serialized card image.

The UPDATE, NUMBER, ======, ENDFIL, REWIND, UNLOAD, PRINT, ENDUP, and SKPFIL pseudo-instructions may also be serialized. If they are, the following occurs:

1. The update input tape is positioned either at a card image with matching serialization or, if there is no matching serialization, between a card image with lower serialization and one with higher serialization.

2. If there is a card image on the update input tape with matching serialization, it is deleted.

3. The pseudo-instruction is interpreted.

If the pseudo-instruction is not serialized, it is interpreted and executed at the current position of the update input tape.

SERIALIZING UNSERIALIZED DECKS

An unserialized deck (or a deck serialized in columns other than 73-80) may be serialized by using the NUMBER pseudo-instruction. The deck to be serialized is placed on the system input tape with the appropriate control cards and update pseudo-instructions. No update input tape is required. The serialized deck will be written on the update output tape.

Caution: File mark (7-8 punched in column 1) cards should not appear in the deck to be serialized. When Update encounters a file mark card on the system input tape, an error message is printed on-line and off-line, and the update job is terminated. Any file mark cards in the deck to be serialized should be replaced with ENDFIL pseudo-instruction cards.

Following is an example of how unserialized decks may be serialized using the NUMBER pseudo-instruction:

```
1       8         16
$JOB
$EXECUTE          UPDATE
        UPDATE    ,9
AR      NUMBER    1
        .
        .         (unserialized deck)
        .
        ENDFIL    (replaces file mark card)
        .
        .         (unserialized deck)
        .
        ENDFIL    (replaces file mark card)
        UNLOAD
        ENDUP
```

The two unserialized decks would be written on the update output tape (logical tape number 9 in this example). They would be blocked 10 cards per block (except for control and END cards) and numbered consecutively in increments of 10 starting with AR000010.

TAPE POSITIONING

To insert card images without serial numbers, the programmer should use a "spacer card" to position the update input tape at the point where the unserialized cards are to be inserted. The spacer card is a duplicate of the card preceding the point at which the unserialized cards are to be inserted. The spacer card replaces the corresponding card on the update input tape. The update input tape is then in the proper position for inserting the unserialized cards which follow the spacer card. Spacer cards may also be used to position tapes before using the ENDFIL or ENDUP pseudo-instructions.

Unserialized cards on an update input tape can only be changed or deleted by deleting the last serialized card in front of the unserialized cards, then deleting all cards up to and including the first serialized card following the unserialized cards, and then reinserting the cards involved. However, if the NUMBER pseudo-instruction is used to serialize insertions when they are made, this cumbersome process can be avoided in future update jobs.

ILLEGIBLE INPUT INSTRUCTIONS

Card images that cause a validity check when read are considered illegible. Such instructions on the system input tape terminate the update job. Illegible instructions on the update input tape are omitted and a message is printed on-line and off-line, but updating continues. Lost instructions may be inserted during a later update job.

The following examples illustrate some uses of the Symbolic Update Program.

The pseudo-instruction

```
UPDATE  9,10
```

merges the correction cards that follow it on the system input tape with those on logical tape 9, the update input tape. A blocked symbolic tape is written on logical tape 10, the update output tape.

The sequence

```
$EXECUTE        UPDATE
        UPDATE  9,10
        CLA     =10                 F0007770
        END     START               F0007990
        ENDFIL
        REWIND
        UNLOAD  9
        ENDUP
```

deletes the card image numbered F0007770 on the update input tape and insert the card image

```
        CLA     =10                 F0007770
```

in its place on the update output tape. If no matching serialization exists on the update input tape, this card will be inserted between cards of lower and higher serialization. Then the spacer card

```
        END     START
```

positions the update input tape at the end of the program it contains. Next, a file mark is written on the update output tape, and that tape is rewound. Finally, logical tape 9, the update input tape, is rewound and unloaded, and updating terminates.

The sequence

```
        UPDATE  ,10
$STOP                               ((((((((
        ENDFIL
        REWIND
        ENDUP
```

at the end of an update job places a $STOP card with serialization of all left parentheses (the highest possible serialization in the binary collating sequence) on the update output tape. The presence of this card prevents tape runaway if, in a later update job, a keypunch or some other error causes Update to search for a serial number higher than any that exists on the update input tape.

Figures 2-5 show the four parts of an update job. Figure 2 is a listing of the program on the update input tape, containing a program to update the update input tape. When the update job is completed, deletions and insertions are listed on the system output tape, Figure 4, and the modified program is on the update output tape, Figure 5. Note that instructions deleted with the IGNORE or SKIPTO pseudo-operations do not appear on the system output tape.

```
1       8       16                                                      73

$IBMAP  READ    M94,NODD
START   RTDA    5                                                       MOV00010
        RCHA    READ                                                    MOV00020
        AXT     9,1                                                     MOV00030
TEST    LDQ     INPUT+9,1                                               MOV00040
        AXT     36,2                                                    MOV00050
SHIFT   PXD     ,0                                                      MOV00060
        LGL     1                                                       MOV00070
        TZE     ZERO                                                    MOV00080
        CLA     =1                                                      MOV00090
        STO     OUTPUT+9,1                                              MOV00100
        TRA     *+2                                                     MOV00110
ZERO    STZ     OUTPUT+9,1                                              MOV00120
        TIX     SHIFT,2,1                                               MOV00130
        TIX     TEST,1,1                                                MOV00140
        CALL    PDUMP(INPUT,OUTPUT+9,0,START,READ,3)                    MOV00150
INPUT   BSS     20                                                      MOV00160
OUTPUT  BSS     9                                                       MOV00170
READ    IORT    INPUT,,20                                               MOV00180
        END     START                                                   MOV00190
```

Figure 2. Update Input Tape

```
1       8       16                                                      73

$IBSYS
$JOB            FIGURE 2
$EXECUTE        UPDATE
        UPDATE  9,3,U
MOV     NUMBER  1,100
*THIS ROUTINE WILL READ RCDS OF UP TO 20 WDS AND WRITE                  MOV00000
*OUT THE FIRST 9 WDS OF EACH RCD.                                       MOV00001
START   REWA    5                                                       MOV00010
        RTDA    5                                                       MOV00011
        TCOA    *                                                       MOV00021
        DELETE                                                          MOV00050
        SKIPTO                                                          MOV00100
        STQ     OUTPUT+9,1                                              MOV00100
        IGNORE                                                          MOV00110
        IGNORE  THRU                                                    MOV00130
        WTDA    3                                                       MOV00141
        RCHA    WRITE                                                   MOV00142
        TCOA    *                                                       MOV00143
        HTR     *                                                       MOV00150
WRITE   IOCD    OUTPUT,,9                                               MOV00181
        END     START                                                   MOV00190
        ENDFIL  3
        REWIND  3
        UNLOAD  9
        ENDUP
$IBSYS
$STOP
```

Figure 3. System Input Tape

```
        UPDATE  9,3,U
MOV     NUMBER  1,100
*THIS ROUTINE WILL READ RCDS OF UP TO 20 WDS AND WRITE    MOV00000 INSERTED
*OUT THE FIRST 9 WDS OF EACH RCD.                         MOV00001 INSERTED
START   RTDA    5                                         MOV00010 DELETED
START   REWA    5                                         MOV00010 INSERTED
        RTDA    5                                         MOV00011 INSERTED
        TCOA    *                                         MOV00021 INSERTED
        AXT     36,2                                      MOV00050 DELETED
        STO     OUTPUT 9,1                                MOV00100 DELETED
        STQ     OUTPUT 9,1                                MOV00100 INSERTED
        WTDA    3                                         MOV00141 INSERTED
        RCHA    WRITE                                     MOV00142 INSERTED
        TCOA    *                                         MOV00143 INSERTED
        CALL    PDUMP(INPUT,OUTPUT,0,START,READ,3)        MOV00150 DELETED
        HTR     *                                         MOV00150 INSERTED
WRITE   IOCD    OUTPUT,,9                                 MOV00181 INSERTED
        END     START                                     MOV00190 DELETED
        END     START                                     MOV00190 INSERTED
```

Figure 4. System Output Tape

```
1       8       16                                                  73

$IBMAP READ     M94,NODD
*THIS ROUTINE WILL READ RCDS OF UP TO 20 WDS AND WRITE              MOV00100
*OUT THE FIRST 9 WDS OF EACH RCD.                                  MOV00200
START   REWA    5                                                  MOV00300
        RTDA    5                                                  MOV00400
        RCHA    READ                                               MOV00500
        TCOA    *                                                  MOV00600
        AXT     9,1                                                MOV00700
TEST    LDQ     INPUT+9,1                                          MOV00800
        STQ     OUTPUT+9,1                                         MOV00900
        TIX     TEST,1,1                                           MOV01000
        WTDA    3                                                  MOV01100
        RCHA    WRITE                                              MOV01200
        TCOA    *                                                  MOV01300
        HTR     *                                                  MOV01400
INPUT   BSS     20                                                 MOV01500
OUTPUT  BSS     9                                                  MOV01600
READ    IORT    INPUT,,20                                          MOV01700
WRITE   IOCD    OUTPUT,,9                                          MOV01800
        END     START                                              MOV01900
```

Figure 5. Update Output Tape

APPENDIX A: TAPE UNIT ASSIGNMENTS

Figure 6 may be used as a guide in selecting units for update input and update output functions.

Tapes 11-16 are selected from the IBSYS Unit Availability Table. In the distributed system, when none of the tapes has been removed from the unit availability chain, the following units are used if requested by Update:

UPDATE Logical Units	Physical Units
11	A6
12	B6
13	A7
14	B7
15	A8
16	B8

The following information is supplied so that Update users may determine which tape units are used when units have been removed from the unit availability chain.

In the distributed system, logical units 11, 13, and 15 select their physical units from Channel A tapes in ascending order in the unit availability chain. Logical units 12, 14, and 16 select their physical units from channel B tapes in ascending order in the unit availability chain.

For example, if unit A6 were removed from the unit availability chain, the following correspondence would exist in the distributed system:

UPDATE Logical Units	Physical Units
11	A7
12	B6
13	A8
14	B7
15	--
16	B8

The user may attach any referable tape units by using $ATTACH and $AS IBSYS control cards. The publication IBM 7090/7094 IBSYS Operating System: System Monitor (IBSYS), Form C28-6248, gives information about these cards.

UPDATE Logical Units	SYSUNI Functions	Physical Units on Distributed Tape	Functions
1	SYSLB1	A1	System Library
2	SYSUT3	A4	Available for updating
3	SYSUT4	B4	Available for updating
4	SYSUT1	A3	Available for updating
5	SYSIN1	A2	System Input
6	SYSOU1	B1	System Output
7	SYSPP1	B2	System Peripheral Punch
8	SYSUT2	B3	Available for updating
9	SYSCK1	NONE	Available for updating
10	SYSCK2	A5	Available for updating
11-16	*	*	Available for updating

Figure 6. Unit Selection Guide

The binary collating sequence places numbers, letters, and special characters in the following order. The octal equivalent of each character is shown in the second column; the corresponding card code for each character is shown in the third column.

Character	BCD Code (Octal)	Card Code
0	00	0
1	01	1
2	02	2
3	03	3
4	04	4
5	05	5
6	06	6
7	07	7
8	10	8
9	11	9
= (equals)	13	8-3
' (apostrophe)	14	8-4
+ (plus)	20	12
A	21	12-1
B	22	12-2
C	23	12-3
D	24	12-4
E	25	12-5
F	26	12-6
G	27	12-7
H	30	12-8
I	31	12-9
. (period)	33	12-8-3
) (right parenthesis)	34	12-8-4
- (minus)	40	11
J	41	11-1
K	42	11-2
L	43	11-3
M	44	11-4
N	45	11-5
O	46	11-6
P	47	11-7
Q	50	11-8
R	51	11-9
$ (dollar sign)	53	11-8-3
* (asterisk)	54	11-8-4
(blank)	60	(blank)
/ (slash)	61	0-1
S	62	0-2
T	63	0-3
U	64	0-4
V	65	0-5
W	66	0-6
X	67	0-7
Y	70	0-8
Z	71	0-9
, (comma)	73	0-8-3
((left parenthesis)	74	0-8-4

Figure 7. Binary Collating Sequence

INDEX

Assembly 5
$* Card 7

Blocking, update output tape 5,9,10

Compilation 5

DELETE pseudo-instruction 12,15

ENDFIL pseudo-instruction 13,15
ENDUP pseudo-instruction 14,15
Equal sign pseudo-instruction 11,15
$EXECUTE Card 6

*FAP card 5
FAP Update 5

$IBSYS Card 7
$ID Card 7
IGNORE pseudo-instruction 12,15
Insert cards 5,9

$JOB Card 6

NODAT option 7

NUMBER pseudo-instruction 10,15

Operation of update, description 9
Operation of update, examples 17

$PAUSE Card 7
PRINT pseudo-instruction 14,15
Programming examples 17

Reserialization 5,9,10,15,16
REWIND pseudo-instruction 13,15
Serialization
  Insert cards 9,16
  Optional, of pseudo-operation cards 15
  Unserialized decks 15,16
  Update input tape 5,9
  Update output tape 10,15,16
Spacer card 13,14,15,16
$STOP Card 7
System input tape 5,9
System output tape 5,7,9

THRU option 12

$TITLE Card 7

UMC pseudo-operation 5
Unit availability 20
UNLOAD pseudo-operation 14,15
Update input tape 5,9
Update logical tape units 20
Update output tape 5
Update pseudo-instructions
  Card Format 9
  DELETE 12,15
  ENDFIL 13,15
  ENDUP 14,15
  IGNORE 12,15
  NUMBER 10,15
  PRINT 14,15
  REWIND 13,15
  SKIPFIL 14,15
  SKIPTO 12,15
  UNLOAD 14,15
  UPDATE 9,15
  Using 9
  ====== 11,15

====== pseudo-instruction 11,15

CUT ALONG LINE

COMMENT SHEET

IBM 7090/7094 IBSYS OPERATING SYSTEM, VERSION 13
SYMBOLIC UPDATE PROGRAM
FORM C28-6386-1

FROM

NAME _____

OFFICE NO. _____

FOLD

CHECK ONE OF THE COMMENTS AND EXPLAIN IN THE SPACE PROVIDED

FOLD

☐ SUGGESTED ADDITION (PAGE)

☐ SUGGESTED DELETION (PAGE)

☐ ERROR (PAGE)

EXPLANATION

FOLD

FOLD

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
FOLD ON TWO LINES, STAPLE, AND MAIL

STAPLE

C28-6386-1

FOLD FOLD

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P. O. BOX 390
POUGHKEEPSIE, N.Y. 12602

ATTN: PROGRAMMING SYSTEMS PUBLICATIONS
DEPARTMENT D39

FOLD FOLD

CUT ALONG LINE

Printed in U.S.A. C28-6386-1

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601

STAPLE